The Montagovian generative lexicon $\Lambda\mathsf{Ty}_n$: an integrated type-theoretical framework for compositional semantics and lexical pragmatics

Christian Retoré
Université de Bordeaux & IRIT-C.N.R.S. & LaBRI-C.N.R.S.
emailchristian.retore@labri.fr
`http://www.labri.fr/perso/retore`

January 22, 2013

Abstract

In the recent years, different but similar approaches integrated lexical semantics within compositional semantics. These approaches all make use of type theory at least to compose the meanings. We present here with some details one of these approaches, ours, which makes use of second order lambda calculus as a type theory for meaning assembly and of multi sorted higher order predicate logic for semantic representation.The advantages of such an approach are illustrated both in formal semantics (determiners, quantifiers, plurals) and in lexical pragmatics (coercions, possible and impossible copredication over different senses, deverbal ambiguities, fictive motion,..). Thereafter we explain the similar approaches by pointing out the differences.

1 Introduction: word meaning and compositional semantics

Semantics is usually divided into formal semantics, usually compositional, which has strong connections with logic and with philosophy of language, and lexical semantics which rather concerns words, derivational morphology and knowledge representation. Roughly speaking formal semantics determines *who does what*, while lexical semantics analyses *what the sentence or discourse speaks about.* Herein we shall endow compositional semantics with a treatment of some of lexical semantics issues, in particular for picking up the right word sense in a given context.

1.1 The syntax of compositional semantics

In this paper we speak about semantics, but as opposed to main stream "formal semantics" in linguistics this paper neither deals with reference nor with truth in a given situation. In the traditional view as exposed in Montague, the process of semantic interpretation of a sentence, consists in computing a logical formula from syntax and word meanings, formula which is interpreted in possible world semantics. Although Montague thought that intermediate steps were meaningless and should be wiped off after computing truth values and references, in this paper we precisely focus on the intermediate step, the logical formula, that can be called the *logical form* of the sentence, and the way it is computed — for the time being, we leave out the interpretation in possible worlds. A reason for doing so is that we can encompass subtle questions, like vague predicates, generalised and vague quantifiers, for which standard notions of truth and references are inadequate possibly some interactive interpretation would be better suited, e.g. like Abrusci and Retoré (2011); Lecomte and Quatrini (2011). Another reason is that, apart from these difficult questions, we do not have modification to bring to standard interpretations.

1.2 Brief reminder on Montague semantics

Let us briefly remind the reader how one computes the logical form according to the montagovian view. Assume for simplicity that a syntactic analysis is a tree specifying for each node, which subtree applies to the other one — the one that is applied is called the function while the other is called its argument. A semantic lexicon provides a simply typed λ-term $[w]$ for each word w. The semantics of a leaf (hence a word) w is $[w]$ and the semantic $[t]$ of a sub syntactic tree $t = (t_1, t_2)$ is recursively defined as $[t] = ([t_1]\ [t_2])$ that is $[t_1]$ *applied to* $[t_2]$, if $[t_1]$ is the function and $[t_2]$ the argument — and as $[t] = ([t_2]\ [t_1])$ otherwise, i.e. when $[t_2]$ is the function and $[t_1]$ the argument.

The typed λ-terms from the lexicon are given in such a way that the function always has a semantic type of the shape $a \to b$ that matches the type a of the argument, and the semantics associated with the whole tree has the semantic type $\mathbf{t}$. This correspondence between syntactical categories and semantic types, which extends into a correspondence between parse structures and logical forms is especially clearin categorial grammars, see e.g. (Moot and Retoré, 2012, Chapter 3). Typed λ-terms usually are defined out of two base types, $\mathbf{e}$ for individuals and $\mathbf{t}$ for propositions. Logical formulae can be defined in this typed λ-calculus as observed by Church long ago. One needs constants for the logical quantifiers and connectives,

Constant	Type
$\exists$	$(\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$
$\forall$	$(\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$

Constant	Type
and	$\mathbf{t} \to (\mathbf{t} \to \mathbf{t})$
or	$\mathbf{t} \to (\mathbf{t} \to \mathbf{t})$
implies	$\mathbf{t} \to (\mathbf{t} \to \mathbf{t})$

word	***semantic type*** u^* ***semantics : λ-term of type*** u^* x^v *the variable or constant* x *is of type* v
some	$(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge^{t\to(t\to t)}(P\ x)(Q\ x))))$
club	$e \to t$ $\lambda x^e(\mathtt{club}^{e\to t}\ x)$
defeated	$e \to (e \to t)$ $\lambda y^e\ \lambda x^e\ ((\mathtt{speak\_about}^{e\to(e\to t)}\ x)y)$
Leeds	e Leeds

Figure 1: A simple semantic lexicon

Constant	Type
$defeated$	$\mathbf{e} \to (\mathbf{e} \to \mathbf{t})$
$won, voted$	$(\mathbf{e} \to \mathbf{t})$
$Liverpool, Leeds$	$\mathbf{e}$

as well as predicates for the precise language to be described — a binary predicate like *speak_about* has the type $\mathbf{e} \to \mathbf{e} \to \mathbf{t}$.

A small example goes as follows. Assume the syntax says that the structure of the sentence "*Some club defeated Leeds.*" is

$$\text{(some (club)) (defeated Leeds)}$$

where the function is always the term on the left. If the semantic terms are as in the lexicon in figure 1, placing the semantical terms in place of the words yields a large λ-term that can be reduced:

$$\Big(\big(\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge(P\ x)(Q\ x))))\big)\big(\lambda x^e(\mathtt{club}^{e\to t}\ x)\big)\Big)$$
$$\Big(\big(\lambda y^e\ \lambda x^e\ ((\mathtt{defeated}^{e\to(e\to t)}\ x)y)\big)\ Leeds^e\Big)$$
$$\downarrow \beta$$
$$\big(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge^{t\to(t\to t)}(\mathtt{club}^{e\to t}\ x)(Q\ x))))\big)$$
$$\big(\lambda x^e\ ((\mathtt{defeated}^{e\to(e\to t)}\ x)Leeds^e)\big)$$
$$\downarrow \beta$$
$$\big(\exists^{(e\to t)\to t}\ (\lambda x^e(\wedge(\mathtt{club}^{e\to t}\ x)((\mathtt{defeated}^{e\to(e\to t)}\ x)Leeds^e)))\big)$$

This λ-term of type $\mathbf{t}$ that can be called the logical form of the sentence, represents the following formula of predicate calculus (admittedly more pleasant to read): $\exists x : e\ (\mathtt{club}(x)\ \wedge\ \mathtt{defeated}(x, Leeds))$.

The above described procedure is quite general: starting a properly defined semantic lexicon whose terms only contains the logical constants and the predicates of the given language one always obtain a logical formula. Indeed, such

λ-terms always reduce to a unique normal form and any normal λ-term of type $\mathbf{t}$ corresponds to a logical formula.

If we closely look at the montagovian setting described above we observe that it is weaving two different "logics":

Logic/calculus for meaning assembly (a.k.a glue logic, metalogic,...) In our example, this is simply typed λ-calculus with two base types $\mathbf{e}$ and $\mathbf{t}$ — these terms are the proof in intuitionistic propositional logic.

Logic/language for semantic representations In our example, that is higher-order predicate logic.[1]

The framework we present in this paper mainly concerns the extension of the metalogic and the reorganisation of the lexicon in order to incorporate some phenomena of lexical semantics in particular restriction of selection. Indeed, in the standard type system above nothing prevents a mismatch between the real nature of the argument and its expected nature. Consider the following sentences:

(1) A chair barks.

(2) Jim ate a departure

(3) The five is fast

Although they can be syntactically analysed, they should not receive a semantical analysis. Indeed, "*barks*" requires a "*dog*" or at least an "*animate*" subject while a "*chair*" is neither of them; "*departure*" is an event, which cannot be an "*inanimate*" object that could be eaten; finally a "*number*" like "*five*" cannot do anything fast — but there are particular contexts in which this can happen and we shall also handles these meaning transfers.

1.3 The need of integrating lexical semantics and pragmatics in formal semantics

In order to block the interpretation of the semantically illformed sentences above, it is quite natural to use *types*, where the word *type* be both understood in its intuitive and in its formal meaning. The type of the subject of barks should be "*dog*", the type of "*fast*" objects should be "*animate*", and the type of the object of "*ate*" should be "*inanimate*". Clearly, having, on the formal side a unique type $\mathbf{e}$ for all entities is not sufficient.

The view with a single type $\mathbf{e}$ for entities has another related drawback. It is unable to link related predicates, although a usual dictionary does. A common noun like "*book*" is viewed as a unary predicate "$\underline{\text{book}}:\mathbf{e} \to \mathbf{t}$" and a transitive verb like "*read*" maps to a binary predicate "$\underline{\text{read}}:\mathbf{e} \to \mathbf{e} \to \mathbf{t}$" This will only give the argument structure of *Mary reads a book.* as $(\exists x :$

[1]It can be first-order logic if reification is used, but this may induce unnatural structure and exclude some readings.

$\mathbf{e}book(x)$ *and* $reads(Mary, x)$) in a such a setting there is no way to relate the predicates <u>book</u> and <u>read</u> — while any dictionary does. If we had several types we could stipulate that the object of read ought to be a book or rather something that can be read "*written*" and that "*book*" objects are part of "*written*" object. A connection between "*read*" and "*book*" would allow to interpret sentences like "*i enjoyed the book*" i.e. "*i enjoying reading the book*".

Hence we need a more sophisticated type theory than the ones initially used by Montague to filter semantically invalid sentences. But in some cases some flexibility is needed to accept and analyse sentences in which a word type is coerced into another type. In sentence 3 "*five*" can be considered as a "*person*" who plays football.

There is a wide literature on such lexical meaning transfers and coercion, starting from 1980 Bierwisch (1979, 1983); Cruse (1986); Nunberg (1995) — see also Lauer (2004); Blutner (2002) for a more recent account of some theories. In those pioneering studies, the objective is mainly to classify these phenomena, to find the rules that govern them. The quest of a computational formalisation that can be incorporated into an automated semantic analyser appears with Pustejovsky's generative lexicon in 1991, Pustejovsky (1991, 1995). The integration of lexical issue into compositional semantics la Montague and type theories appears with the work by Nicholas Asher Asher and Pustejovsky (2000); Asher (2008) which lead to the book Asher (2011).

We must mention the work of Cooper which, also it uses type theory as well, seem rather different from the ones we are going to study. Indeed, via tan intensive use of records from type theory it is closer to frame semantics, with features and attributes. Cooper (2007, 2011) Hence, despite the interest of this work it is hard to to draw a comparison, with the work we are going to describe.

1.4 Lexical and compositional semantics

As the work of Asher suggests, finer-grained type theories are quite a natural framework both for formal semantics in montagovian style and for selectional restriction and coercions. Such a model must extend the usual ones into two directions:

1. Montague's original type system and metalogic should be enriched to encompass lexical issues (selectional restriction and coercions), and

2. the usual constructs in formal semantics should be extended to this richer type system (which is rarely done except Cooper (2007, 2011); Chatzikyriakidis and Luo (2012); Moot and Retoré (2011); Lefeuvre et al. (2012b); Retoré (2012)

At the end of this paper, we shall provide a comparison of the current approaches, which mainly focus on 1. Let us list right now what are the current approaches:

- The system work with type based coercions and relies on some *Modern*

Type Theory (MTT) [2] — this correspond to the work of Zhaohui Luo Luo (2011, 2012); Xue and Luo (2012); Chatzikyriakidis and Luo (2012)

- The system work with type based coercions and relies on categorical variant of usual typed λ-calculus — this approach by Asher Asher and Pustejovsky (2000); Asher (2008) culminated in his book Asher (2011)
- The system work with term based coercions and relies on second order λ-calculus — this is our approach, developed with Bassac, Mery, Moot, Real-Coelho. Bassac et al. (2010); Moot et al. (2011b,a); Moot and Retoré (2011); Lefeuvre et al. (2012a,b); Retoré (2012); Real-Coelho and Retoré (2013)

Rather than presenting all these approaches, we shall present ours and thereafter comment on the differences. Our approach has the particularity that we studied some compositional semantics phenomena in this framework firstly designed for lexical matters — although Asher and Luo also worked in this direction Chatzikyriakidis and Luo (2012); Asher (2011).

In fact our approach differs mainly because of the organisation of the lexicon and of the respective rôles of types and terms. The precise type system we use, namely system F , does not mak a big difference, and as far as the presentation of the system is concerned, it is the simplest of all systems, because it only contains four term building operations (two of them being the standard λ-calculus rules, the two other one being their second order counter part) and two reduction rules (one of them being the usual beta reduction and the other one being its second order counterpart).

2 A Montagovian generative lexicon for compositional semantic and lexical pragmatics

We are to present our solution for introducing some lexical issues in a compositional framework à la Montague. We should keep in mind that whatever the precise solution presented, the following questions must be addressed in order to obtain a computational model, so here are the general principles guiding our model:

- What is the logic for semantic representation?
  We use many-sorted higher order predicate calculus. As usual, the higher order can be reified in first order logic, so it can be first order logic, but in any case the logic has to be many sorted. Asher Asher (2011) is quite similar , while Luo use Type Theory Luo (2012).

[2]This name *Modern Type Theory (MTT)* covers several variants of modern type theories, including Martin-Löf type theory, the Predicative Calculus of (Co)Inductive Constructions (pCic), the Unifying Theory of dependent Types (UTT),... — this later one being the closest to the system used by Zhaohui Luo

- What is the metalogic (glue logic) for meaning assembly?
  We use second order λ-calculus (Girard system F) in order to factor operations that apply uniformly to family types. Asher Asher (2011) use simply typed λ-terms with additional categorical rules, while Luo also use Type Theory with coercive sub typing Luo (2012).

- What kind of information is associated with a word in the lexicon?
  Here it will be a finite set of λ-terms, one of them being called the principal λ-term while the other are called optional. Other approaches make use more specific terms and rules.

- How does one compose words and constituents for a compositional semantics?
  We simply apply one λ-term to the other, following the syntactic analysis, perform some transformations corresponding to coercions and presupposition, and reduce the compound by β-reduction.

- How is rendered the semantic incompatibility of two components?
  By type mismatch, between a function of type $A \to X$ and an argument of type $B \neq A$, and others do the same.

- How does one allow an *a priori* impossible composition?
  By using the optional λ-terms, which change the types of at least one of the two terms, function and argument. Other approaches rather use type driven rules.

Each word in the lexicon is given a principal term, as well as a finite number, possibly nought, optional terms that licence type change and implement coercions. They may be inferred from an ordinary dictionary, electronic or not. They combine almost as usual except that there might be type clashes, which accounts for infringement of selectional restriction: in this case optional terms may be use to solve the type mismatch. In case they lead to different results these results should be considered as different possible readings — e.g. as readings with different quantifier scopes are considered by formal semantics as different possible readings of a sentence.

Let us first present the type and terms and thereafter we shall come back to the the composition modes.

2.1 Many sorted logic in λ-calculus

We use a type system that resembles Muskens Ty_n Muskens (1990), that is the type of individuals, $\mathbf{e}$ is replaced with a finite but large set of base types $\mathbf{e}_1, \ldots, \mathbf{e}_n$ for individuals, for instance objects, concepts, events,... They are the sorts of the many sorted logic whose formulae express semantic representations. For instance, assume we have a many sorted logic with a sort ζ for animals, a sort ϕ for physical objects and a predicate *eat* whose arguments are of respective sort ϕ and ζ the many sorted formula $\forall z : \zeta\ \exists x : \phi\ eat(z,x)$ is rendered in type theory by the λ-term: $\forall^{\zeta}(\lambda z^{\zeta}\exists^{\phi}\lambda x^{\phi}((eat\ x)z)$ with *eat* a constant of type $\phi \to \zeta \to \mathbf{t}$ — observe that the type theoretic formulation requires a quantifier for each sort α of object, that is a constant $\forall^{\alpha}$ of type $(\alpha \to \mathbf{t}) \to \mathbf{t}$. [3]

What are the base types? We want to leave this choice as open as it may, without being unrigorous. Indeed, this is a subtle question depending on ones philosophical convictions, but it does not really interfere with the formal and computational model we present here. Let us mention some natural sets of bases types are, from the smallest to the largest:

1. a single type **e** for all entities (but as seen above it cannot account for lexical semantics)

2. a very simple ontology distinguishing events, physical objects, living entities, concepts, ... (this resembles Asher's position)

3. a type per common noun (that's the solution of Luo in Luo (2012))

4. a type for every formula with a single free variable

Our opinion is that types should be cognitively natural classes and rich enough to express selectional restrictions. Whatever types are, there is a relation between types and properties. With base types as in 4, the correspondence seems quite clear, but, because types can be used to express new many sorted formulae, the set of types is in this case defined as a least fixed point. For other and smaller set of types, e.g. 3 or 2 for each type τ there should be a corresponding predicate which recognises τ entities among entities of a larger type. For instance, if there is a type "*dog*" there should be a predicate $\widehat{\text{"}dog\text{"}} : \alpha \to \mathbf{t}$ but what should be α the type of its argument? Should it be "*animal*", "*animate*",... the simplest solution is to assume a type of all individuals, that is Montague's **e**, and to say that corresponding to any base type τ, there is a predicate, namely $\widehat{(\tau)}$ of type $\mathbf{e} \to \mathbf{t}$[4]

Let us say here a remark on the predicate constants in the language. If a predicate constant, say Q is given with type $u \to \mathbf{t}$ with $u \neq \mathbf{e}$ which sometimes is more natural there is an obvious extension Q_e which should be interpreted as false for any object that cannot be viewed as an u-object. Given predicate in the language do also have restrictions, $Q|_v$ which is defined as Q on $q \cap v$ where q is the domain of Q and false elsewhere.

2.2 $\Lambda\mathsf{Ty}_n$: many sorted formulae in second order λ-calculus (Girard's system F)

Since we have many base types, and many compound types as well, it is quite convenient and almost necessary to define operations over family of similar terms

[3]We do not speak about interpretations, but if one wishes to, we do not necessarily ask for the usual requirement that sorts are disjoint, – in type theory, nothing prevents a term to have several types.

[4]An alternative solution would be $\Pi\alpha.\ \alpha \to \mathbf{t}$, using quantification over types to be defined in next section.

with different types, to have some flexibility in the typing, and to have terms that act upon types. Hence we shall extend further Ty_n into $\Lambda\mathsf{Ty}_n$ by using Girard's system F as the type system Girard (2011, 1971). System F involves quantified types whose terms can be specialised to any type.

- Constants $\mathbf{e}_i$ and $\mathbf{t}$, as well as any type variable α are types.
- Whenever T is a type and α a type variable which may but need not occur in T, $\Pi\alpha.\ T$ is a type.
- Whenever T_1 and T_2 are types, $T_1 \to T_2$ is a type as well.

Terms encode proofs of quantified propositional formulae:

- A variable of type T i.e. $x : T$ or x^T is a *term*, and there are countably many variables of each type.
- $(f\ \tau)$ is a term of type U whenever $\tau : T$ and $f : T \to U$.
- $\lambda x^T.\ \tau$ is a term of type $T \to U$ whenever $x : T$, and $\tau : U$.
- $\tau\{U\}$ is a term of type $T[U/\alpha]$ whenever $\tau : \Lambda\alpha.\ T$, and U is a type.
- $\Lambda\alpha.\tau$ is a term of type $\Pi\alpha.T$ whenever α is a type variable, and $\tau : T$ a term without any free occurrence of the type variable α in the type of a free variable of τ.

The later restriction is the usual one on the proof rule for quantification in propositional logic: one should not conclude that $F[p]$ holds for any proposition p when assuming $G[p]$ — i.e. having a free hypothesis of type $G[p]$.

The reduction is defined by two schemes which are quite similar:

- $(\lambda x.\tau)u$ reduces to $\tau[u/x]$ (usual β reduction).
- $(\Lambda\alpha.\tau)\{U\}$ reduces to $\tau[U/\alpha]$ (remember that α and U are types).

As an example, we earlier said that in Ty_n we needed a first order quantifier per sort (or base type). Here, a single quantifier $\forall$ of type $\Pi\alpha.\ (\alpha \to \mathbf{t}) \to \mathbf{t}$ is sufficient. Indeed, this quantifier can be specialised to specific types, for instance to the base type ζ, yielding $\forall\{\zeta\} : (\zeta \to \mathbf{t}) \to \mathbf{t}$, or even to properties of ζ objects, which are of type $\zeta \to \mathbf{t}$, yielding $\forall\{\zeta \to \mathbf{t}\} : ((\zeta \to \mathbf{t}) \to \mathbf{t}) \to \mathbf{t}$.

As Girard showed Girard (2011, 1971) reduction is strongly normalising and confluent *every term of every type admits a unique normal form which is reached no matter how one proceeds.* [5] This has a good consequence for us, see e.g. (Moot and Retoré, 2012, Chapter 3):

[5]This is one way to be convinced of the soundness of F , which defines types depending on other types including themselves: as it is easily observed that there are no normal closed terms of type $\Pi X.\ X \equiv \bot$ the system is necessarily coherent. Another way is to construct a concrete model, called coherence spaces, where types are interpreted as countable sets, and terms up to normalisation are interpreted as functions. Girard (2011)

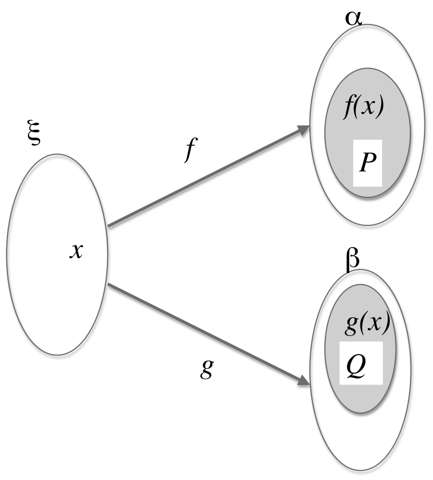

Figure 2: Polymorphic conjunction: $P(f(x))\&Q(g(x))$ with $x:\xi$, $f:\xi\rightarrow\alpha$, $g:\xi\rightarrow\beta$.

Property 1 ($\Lambda\mathsf{Ty}_n$terms as formulae of a many-sorted logic) *If the predicates, the constants and the logical connectives and quantifiers are the ones from a many sorted logic of order n (possibly $n=\omega$) then the normal terms of $\Lambda\mathsf{Ty}_n$ of type $\mathbf{t}$ unambiguously correspond to many sorted formulae of order n.*

Let us illustrate how F factors uniform behaviours. Given types α, β, two predicates $P^{\alpha\rightarrow\mathbf{t}}$, $Q^{\beta\rightarrow\mathbf{t}}$, over entities of respective kinds α and β for any ξ with two morphisms from ξ to α and to β, see figure 2.2 F contains a term that can coordinate the properties P,Q of (the two images of) an entity of type ξ, every time we are in a situation to do so — i.e. when the lexicon provides the morphisms.

Term 1 (Polymorphic AND) *is defined as* $\&^{\Pi}=$ $\Lambda\alpha\Lambda\beta\lambda P^{\alpha\rightarrow\mathbf{t}}\lambda Q^{\beta\rightarrow\mathbf{t}}\Lambda\xi\lambda x^{\xi}\lambda f^{\xi\rightarrow\alpha}\lambda g^{\xi\rightarrow\beta}.\ (and^{\mathbf{t}\rightarrow\mathbf{t}\rightarrow\mathbf{t}}\ (P\ (f\ x))(Q\ (g\ x)))$

This can apply to say, a book, that can be heavy as a physical object, and interesting as an informational content — the limitation of possible over generation is handled by the *rigid* use of possible transformations, including identity to be defined thereafter.

2.3 Organisation of the lexicon and rules for meaning assembly

The lexicon associate each word w with a principal λ-term $[w]$ which basically is the Montague term reminded earlier, except that the types appearing in $[w]$

word	principal λ-term	optional λ-terms	rigid/flexible
book	$\widehat{B}:\mathbf{e}\to\mathbf{t}$	$Id_B : B \to B$	(F)
		$b_1 : B \to \phi$	(F)
		$b_2 : B \to I$	(F)
town	$\widehat{T}:\mathbf{e}\to\mathbf{t}$	$Id_T : T \to T$	(F)
		$t_1 : T \to F$	(R)
		$t_2 : T \to P$	(F)
		$t_3 : T \to Pl$	(F)
Liverpool	$liverpool^T$	$Id_T : T \to T$	(F)
		$t_1 : T \to F$	(R)
		$t_2 : T \to P$	(F)
		$t_3 : T \to Pl$	(F)
wide	$wide : Pl \to \mathbf{t}$		
voted	$voted : P \to \mathbf{t}$		
won	$won : F \to \mathbf{t}$		

where the base types are defined as follows:

ϕ	physical objects	I	information	P	people
B	book	T	town	Pl	place

Figure 3: A sample lexicon

belong to a much richer system. In particular they can impose some selectional restriction. In addition, there can be optional λ-terms to allow, in some cases, composition that were initially ruled out by selectional restriction.In addition, there are two ways to solve a type conflict: every optional λ-terms is declared, in the lexicon, to be a rigid modifier, noted (R) or a flexible one, noted (F). Rigid modifiers turn the type, or the sense of a word, into another one which is incompatible with other types or senses. For a technical reason, the identity which is always a licit modifier is also specified to be flexible or rigid. In this later rigid case, it means that the original sense is incompatible with derived senses.

The reader may be surprised that we repeat the morphisms in the lexical entries, rather than having general rules. One could also consider morphisms that are not anchored in a particular entry: in particular, they could implement the ontology at work in Pustejovsky (1995). For instance, a place (type Pl) could be viewed as a physical object (type ϕ) with a general morphism $P2\phi$ turning places into physical objects that can be "*wide*". We are not enthusiastic about using such general rules since it is hard to tell whether they are flexible or rigid. As they can be composed they might lead to incorrect copredications, while their repetition inside each entry offers a better control of incorrect and correct copredications. One can think that some meaning transfer differs although the words have the same type. An example could be in French the words "*classe*" and "*promotion*", which are both groups of human beings, students or pupils.

The first word "*classe*" can be coerced into a room, the room where the pupils are taught, while the second, "*promotion*", cannot.

One can foresee what is going to happen, using the lexicon given in figure 3 with sentences like:

(4) Liverpool is wide.

(5) Liverpool is wide and voted (last Sunday).

(6) # Liverpool voted and won (last Sunday).

Our purpose is not discuss whether this or that sentence is correct, nor whether this or that copredication is felicitous, but to provide a formal and computational model which given sentences that are assumed to be correct, derives the correct readings, and which given sentences that are said to be incorrect, fails to provide a reading.

Ex. 4 This sentence leads to a type mismatch $wide^{Pl\to\mathbf{t}}(Liverpool^{T}))$, since "*wide*" applies to "*places*" (type Pl) and not to "*towns*" as "*Liverpool*". It is solved using the optional term $t_3^{T\to Pl}$ provided by the entry for "*Liverpool*", which turns a town (T) into a place (Pl) $wide^{Pl\to\mathbf{t}}(t_3^{T\to Pl} Liverpool^{T}))$ — a single optional term is used, the (F)/ (R)difference is useless.

Ex. 5 In the second example, the fact that Liverpool is wide is derived as previously, and the fact Liverpool voted is obtained from the transformation of the town into people, that can vote. The two can be conjoined by the polymorphic "*and*" defined above as term **??** ($\&^{\Pi}$) because these transformations are flexible: one can use one and the other. We can make this precise using only the rules of the type calculus. The syntax yields the predicate $(\&^{\Pi}(is\_wide)^{Pl\to\mathbf{t}}(voted)^{P\to\mathbf{t}})$ and consequently the type variables should be instantiated by $\alpha := Pl$ and $\beta := P$ and the exact term is $\&^{\Pi}\{Pl\}\{P\}(is\_wide)^{Pl\to\mathbf{t}}(voted)^{P\to\mathbf{t}}$ which reduces to:
$\Lambda\xi\lambda x^{\xi}\ \lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}(\text{and}^{\mathbf{t}\to\mathbf{t})\to\mathbf{t}}\ (is\_wide\ (f\ x))(voted\ (g\ x)))$.
Syntax also says this term is applied to "*Liverpool*". which forces the instantiation $\xi := T$ and the term corresponding to the sentence is after some reduction steps,
$\lambda f^{T\to Pl}\lambda g^{T\to P}(\text{and}\ \ (is\_wide\ \ (f\ \ Liverpool^{T}))(voted\ \ (g\ \ Liverpool^{T}))))$.
Fortunately the optional λ-terms $t_2 : T \to P$ and $t_3 : T \to Pl$ are provided by the lexicon, and they can both be used, since none of them is rigid. Thus we obtain, as expected
$(\text{and}\ (is\_widePl \to \mathbf{t}\ (t_3^{T\to Pl}\ Liverpool^{T}))(voted^{Pl\to\mathbf{t}}\ (t_2^{T\to P}\ Liverpool^{T})))$

Ex. 6 The third example is rejected as expected. Indeed, the transformation of the town into a football club prevents any other transformation (even the identity) to be used in the polymorphic and that we defined above. We obtain the same term as above, with *won* instead of *is_wide*. The term is: $\lambda f^{T\to Pl}\lambda g^{T\to P}(\text{and}\ \ (won\ \ (f\ \ Liverpool^{T}))(voted\ \ (g\ \ Liverpool^{T}))))$ and the lexicon provides the two morphisms that would solve the type conflict, but one of them is rigid, i.e. we can solely use this one. Consequently the sentence is semantically invalid.

This also applies to deverbal ambiguity between process and result as Real-Coelho and Retoré (2013) shows. A rather innovative extension is to apply this technique to what Talmy called *fictive motion* Talmy (1999). Under certain circumstances, a path may introduce a virtual traveller following the path, as in sentences like:

(7) Path GR3 descends for two hours.

Because of the duration one cannot consider that the vertical coordinate decreases. One needs to consider someone that follow the road. We model this by one morphism associated with the "*Path GR3*" and one with "*descends*". The first coercion turns the "*Path GR3*" from an immobile object into an object of type "*path*" that can be followed and the second one coerce "*descends*" into a verb that acts upon a "*path*" object and introduce an individual following the path downwards — this individual, which does not need to exist, is quantified, yielding a proposition that can be paraphrased as "*any individual following the path goes downwards for two hours*". Moot et al. (2011b,a)

The examples presented so far only involved proper names because the determiners and quantifiers are a bit more complex than in the usual montagovian setting, let us see how they work.

2.4 Formal semantics questions: determiners, quantification and plurals

In order to integrate lexical pragmatics into compositional semantics which closely follows syntax, we should at least describe the behaviour of determiners and quantifiers in our framework. We adopt the view of quantified, definite, and indefinite noun phrases as *individual terms* by using generic elements (or choice functions) as initiated by Russell and formalised by Hilbert, Ackerman and Bernays see e.g. Hilbert and Bernays (1939) and adapted to linguistics by researchers like von Heusinger see e.g. Egli and von Heusinger (1995); von Heusinger (1997, 2004).

How do we adapt our model, in particular the typing, if instead of "*Liverpool*" the examples used "*The town*", "*A town*", "*All towns*", or "*Most towns*"? Indefinite determiners, quantifiers, generalised quantifiers,... usually are viewed as functions from two predicates to propositions, one expressing the restriction and the other the main predicate see e.g. Peters and Westerståhl (2006)

In accordance with syntax, we prefer to consider that a quantified noun phrase is by itself some individual — a generic one which does not refer to a precise individual nor to a collection of individuals. As von Heusinger (1997) we use a η for indefinite determiners (which pick up new element) and ι for definite noun phrases (which pick up the most salient — von Heusinger (1997) writes ϵ, but we also use Hilbert's ϵ), Hilbert's ϵ and τ, and others for generalised quantifiers. All those operators takes as arguments a predicate $P(\_)$ and return a term, which for ι is written as the term $\iota x.\ P(x)$ in which the variable x is bound — the behaviour of the other generic elements introduced by $\epsilon, \tau, \eta, ...$ is just the same. The main problem is to provide a proper typing of such operators which fits in our model. [6] In a typed model, a predicate applying to α-objects is of type $\alpha \to \mathbf{t}$. Consequently ι should be of type: $(\alpha \to \mathbf{t}) \to \alpha$, and in order to have a single ι its type is $\Pi\alpha.\ (\alpha \to \mathbf{t}) \to \alpha$. Consequently, if we have a predicate "*Dog*" of "*Animate*" entities the term $\iota(Dog)$ (written $\iota x.\ Dog(x)$ in untyped models) the semantics of "*the dog*" is of type "*Animate*".... but we would like this term to be of type *Dog* if "*dog*" is a type, or to enjoy the property *Dog*, if *Dog* is a property. How do we say so, since the type *Dog* does not appear in ι? Indeed, only "*animate*" objects appear in ι as an instantiation of α. We solve this by adding a systematic presupposition that can be called an axiom, $P(\iota(P))$ for any P of type $\mathbf{e} \to \mathbf{t}$ [7]

The syntax of quantifiers and generalised quantifiers is defined in the same way. Existential quantification "*some*" is faithfully represented by Hilbert's *epsilon* operator: $P(\epsilon x P(x)) \equiv \exists x P(x)$. As soon as some element enjoys the property P, the term $\epsilon x P(x)$ enjoys P as well. Regarding τ which symmetrically constructs the generic element that appear in mathematical proofs like *Let x be any integer ...* it works just the same: $P(\tau x P(x)) \equiv \tau x P(x)$: as soon as the term $\tau x P(x)$ enjoys the property P any element does. These terms are typed just the same way, and this construct can be applied to compute the logical form of statement including the "*most*" quantifier, as exposed in Retoré (2012).

The organisation of the types also allows us to handle simple facts about plurals, as shown in Moot and Retoré (2011) — which resembles some Partee's ideas of Partee (2008). Here are some classical examples involving plurals, exemplifying some typical readings for plurals:

(8) The three of us moved the piano.
(collective reading is likely, or but a covering one is possible as well)

(9) The twelve students passed the exam. (each of them)

(10) The committees met. (each committee met or they all met)

Such readings are derivable in our model because one can define in F operators for handling plurals. Firstly, on can add, as a constant, a cardinality operator for predicates $||\_|| : \Pi\alpha.(\alpha \to \mathbf{t}) \to \mathbb{N}$ (using the internal integers of system F which are $\mathbb{N} = \Pi X.\ (X \to X) \to (X \to X)$, or predefined integers as in Gödel system T or most type theories). Next, as shown in figure 4 , we can have operators for handling plurals: q (turning an individual into a property/set), $*$ (distributivity) # (restricted distributivity from sets of sets to its constituent subsets), c (for coverings)...

2.5 Variants and implementation

In the afore presented model, some points admit slight changes that do not affect the behaviour.

[6] Actually, we first provided a type theoretical model,and then discovered earlier related work in untyped semantics, e.g. papers by Heusinger.

[7] If the predicate P corresponds to a type τ i.e. $P = \widehat{\tau}$, this presupposition is better written as $\iota(\widehat{(\tau)}) : \tau$.

$$
\begin{array}{ll}
q & \Lambda\alpha\lambda x^{\alpha}\lambda y^{\alpha} x = y \\
* & \Lambda\alpha\lambda P^{\alpha\to\mathbf{t}}\lambda Q^{\alpha\to\mathbf{t}}\forall x^{\alpha} Q(x) \Rightarrow P(x) \\
\# & \Lambda\alpha\lambda R^{(\alpha\to\mathbf{t})\to\mathbf{t}}\lambda S^{\alpha\to\mathbf{t})\to\mathbf{t}}\forall P^{\alpha\to\mathbf{t}} S(P) \Rightarrow R(P) \\
c & \Lambda\alpha\lambda R^{(\alpha\to\mathbf{t})\to\mathbf{t}}\lambda P^{\alpha\to\mathbf{t}}\forall x^{\alpha} P(x) \Rightarrow \exists Q^{\alpha\to\mathbf{t}} Q(x) \wedge (\forall y^{\alpha} Q(y) \Rightarrow P(y)) \wedge R(Q)
\end{array}
$$

Figure 4: Operators for plurals

Base types can vary from a very restricted set to the infinite set of formulae with a single free variables. In relation to this issue, the inclusion between base types, that could be called ontological inclusions, that in our model are morphisms can be introduced with words, as we did or as general axioms. We prefer the first solution which allows idiosyncratic behaviours, dependent on words. For instance, it is fairly possible that only one of two synonymical words allows the object to be considered with a different type, as we saw in paragraph 2.3 with "*classe*" and "*promotion*". The type we gave for predicate can also vary: it could be systematically $\mathbf{e} \to \mathbf{t}$, but as explained in paragraph 2.4, types $u \to \mathbf{t}$ are possible as well, and varying from one form to another is not complicated.

An important variant is to define the very same ideas within a compositional model like λ-DRT. Thus one can integrate the semantical and lexical issues presented here into a broader perspective. This can be done, and in fact several applications of the model presented here are already included into the Grail parser by Richard Moot, in particular for French Moot (2010). The grammar is an automatically acquired grammar but unfortunately the refined semantic terms we need can only be typed by hand. Consequently we only can test the semantic analyses described herein on small lexicon. The adaptation from the montagovian style used in this paper and its λ-DRT version can be seen for our treatment of fictive motion, first described la Montague Moot et al. (2011b) and thereafter implemented in λ-DRT Moot et al. (2011a)

3 Comparison with related work and conclusion

If we compare our work with the one of Asher and Luo, we should first the reader that there are much more resemblances then differences, this is why we focused on one of the systems based on type theory and integrating some lexical pragmatic issues.

Our type system, F , is quite powerful but simple: four-term building operations, and two reduction rules. Luo make use of a version of Modern Type Theories (MTT), closed to the Unifying Theory of dependent Types (UTT), whose expressive power and computational complexity is more limited, and thus the type theory he uses better characterises the logic needed for meaning assembly. Quantification over type variable is quite comparable and allows $\forall\alpha : CN$ which is quite convenient although it can certainly be encoded within system F using the fact that finites sums can be defined in system F , hence $x : \alpha, \alpha : CN$ can be rephrased if there are finitely many CN — finite products can be fined as well. This is both a positive and negative feature of system F : it can encode many things, but encodings are often dull. In addition, the MTT that Luo uses, includes dependent types, i.e. types defined from terms, which are convenient — the way they are used so far can probably be encoded in system F , but encoding can be tedious. Regarding coercions, Luo makes an extensive use of coercive sub typing, that he introduced with Soloviev Soloviev and Luo (2000): it is likely that this kind of subtyping may also work well with system F . So we can say that Luo system is very similar. It better corresponds to the semantic operations, its expressive power is more limited, but the formal diversity of the many rules may result in an opaque formalisation. The typed system at work in Asher's view is a simple type theory extended with type constructs and operations from category theory. The theory extends cartesian closed category with a few of the many operations that one finds in a topos, like subtype. This approach is hardly compared with the two above, since it does not belong to the same family: morphisms do not represents (quotiented) proofs of some logic, they are closer to a set theoretic interpretation.

Another ingredient of our models are the base types. As we do, Asher leaves the set of base types open: $\mathbf{e}, \mathbf{t}$, physical object, etc., with a linguistically motivated subtyping relation $\sqsubset$ defined over these types. Luo, especially in his later article Luo (2012), wants to equate base types with common nouns (also with coercions between them), and this seems to us a good compromise between any formula and the minimal base type system which makes it difficult to express some selectional restrictions with types. The sub typing relation between the base types are language independent in these two models, i.e. they are not triggered by words, but simply by types. Although we could do the same, this is quite a difference, that is, possibly a word for "*car*" allows to consider a "*car*" as a "*vehicle*", while another word would not.

Regarding the general organisation of the lexicon and of its composition modes, the same difference applies. While according to Asher and Luo the types determine the coercions, in our approach the coercions are provided by the terms in the lexicon, i.e. by the words themselves and not by their types. The recent claim by Luo that base type should be common nouns (that are words) partly rubs out the differences between on one hand the type driven approaches of himself and Asher and, on the other hand, ours which is more idiosyncratic being based on words and terms.

Finally one may wonder whether we finally derive similar logical forms? They actually are quite similar: we derive higher order multi sorted logical formulae multi sorted, Asher derives formulae in an intuitionnistic set theory, which works with sorts, and Luo derives formulae of type theory. All these are more or less the same: higher order is possible although not extensively used in examples, and there are sorts or types.

A possible difference may lie in the distance with syntax and the effective computability of the semantic representation, which requires a treatment of the current constructs in compositional semantics, like determiners, quantifiers, plurals,... and to be integrated in a general analyse also including phenomena like time or aspect. For the time being we did more on such issues than the others, but I am sure similar treatment is possible within the other approaches by Asher and Luo.

We think that the recent series of articles on the integration of lexical pragmatics into type theoretic models of compositional semantics reached a certain maturity and converges despite some slight differences. We hope that the reader will be convinced of the interest of the present state of the art and of the questions that remain open: what should be the base types, what type theory is better suited, how to account for idiosyncratic lexical phenomena, what is adapted notion of sub typing, what is the relation between word knowledge and its lexical representation,... How is the discourse context represented? How does it trigger particular coercions? How can the model account for this contextual triggering? This work also demands a precise linguistic study of the phenomena that we can encode: this should lead to very interesting study of particular phenomena and cross linguistic comparisons, e.g. on selectional restrictions, on deverbals, on facets (or dot types), on the expression of time and space,... It also raises questions of formal semantics within a more refined framework based on type theory: how do determiners, quantification, and generic object acts on sorts?

References

Abrusci, V. M. and Retoré, C. (2011). Quantification in ordinary language: from a critic of set-theoretic approaches to a proof-theoretic proposal. In Schröder-Heister, P., editor, *14th Congress of Logic, Methodology and Philosophy of Sciences.*

Asher, N. (2008). A type driven theory of predication with complex types. *Fundamenta Informaticae*, 84(2):151–183.

Asher, N. (2011). *Lexical Meaning in context – a web of words.* Cambridge University press.

Asher, N. and Pustejovsky, J. (2000). The metaphysics of words in contexts.

Bassac, C., Mery, B., and Retoré, C. (2010). Towards a Type-Theoretical Account of Lexical Semantics. *Journal of Logic Language and Information*, 19(2):229–245. http://hal.inria.fr/inria-00408308/.

Béchet, D. and Dikovsky, A. J., editors (2012). *Logical Aspects of Computational Linguistics - 7th International Conference, LACL 2012, Nantes, France, July 2-4, 2012. Proceedings*, volume 7351 of *Lecture Notes in Computer Science.* Springer.

Bierwisch, M. (1979). Wörtliche bedeutung - eine pragmatische gretchenfrage. In Grewendorf, G., editor, *Sprechakttheorie und Semantik*, pages 119–148. Surkamp, Frankfurt.

Bierwisch, M. (1983). Semantische und konzeptuelle repräsentation lexikalischer einheiten. In Růžička, R. and Motsch, W., editors, *Untersuchungen zur Semantik*, pages 61–99. Akademie-Verlag, Berlin.

Blutner, R. (2002). Lexical semantics and pragmatics. In Hamm, F. and Zimmermann, T. E., editors, *Semantics*, volume 10 (Sonderheft), pages 27–58, Hamburg. Buske.

Chatzikyriakidis, S. and Luo, Z. (2012). An account of natural language coordination in type theory with coercive subtyping. In Parmentier, Y., editor, *Constraint Solving and Language Processing.*

Cooper, R. (2007). Copredication, dynamic generalized quantification and lexical innovation by coercion. In *Fourth International Workshop on Generative Approaches to the Lexicon.* Université de Genève.

Cooper, R. (2011). Copredication, quantification and frames. In Pogodalla and Prost (2011), pages 64–79.

Cruse, D. (1986). *Lexical semantics.* Cambridge textbooks in linguistics. Cambridge University Press.

Egli, U. and von Heusinger, K. (1995). The epsilon operator and E-type pronouns. In Egli, U., Pause, P. E., Schwarze, C., von Stechow, A., and Wienold, G., editors, *Lexical Knowledge in the Organization of Language*, pages 121–141. Benjamins.

Girard, J.-Y. (1971). Une extension de l'interprétation de Gödel à l'analyse et son application: l'élimination des coupures dans l'analyse et la théorie des types. In Fenstad, J. E., editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 63–92, Amsterdam. North Holland.

Girard, J.-Y. (2011). *The blind spot – lectures on logic.* European Mathematical Society.

Hilbert, D. and Bernays, P. (1939). *Grundlagen der Mathematik. Bd. 2.* Springer. Traduction française de F. Gaillard, E. Guillaume et M. Guillaume, L'Harmattan, 2001.

Lauer, S. (2004). A comparative study of current theories of polysemy in formal semantics. Master's thesis, Cognitive science Osnabrück - Computational Linguistics.

Lecomte, A. and Quatrini, M. (2011). Figures of dialogue: a view from ludics. *Synthese*, 183:59–85.

Lefeuvre, A., Moot, R., and Retoré, C. (2012a). Traitement automatique d'un corpus de récits de voyages pyrénéens : analyse syntaxique, sémantique et pragmatique dans le cadre de la théorie des types. In *Congrès mondial de linguistique française.*

Lefeuvre, A., Moot, R., Retoré, C., and Sandillon-Rezer, N.-F. (2012b). Traitement automatique sur corpus de récits de voyages pyrénéens : Une analyse syntaxique, sémantique et temporelle. In *Traitement Automatique du Langage Naturel, TALN'2012*.

Luo, Z. (2011). Contextual analysis of word meanings in type-theoretical semantics. In Pogodalla and Prost (2011), pages 159–174.

Luo, Z. (2012). Common nouns as types. In Béchet and Dikovsky (2012), pages 173–185.

Moot, R. (2010). Wide-coverage French syntax and semantics using Grail. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Montreal.

Moot, R., Prévot, L., and Retoré, C. (2011a). A discursive analysis of itineraries in an historical and regional corpus of travels. In *Constraints in discourse*, page http://passage.inria.fr/cid2011/doku.php, Ayay-roches-rouges, France.

Moot, R., Prévot, L., and Retoré, C. (2011b). Un calcul de termes typés pour la pragmatique lexicale — chemins et voyageurs fictifs dans un corpus de récits de voyages. In *Traitement Automatique du Langage Naturel, TALN 2011*, pages 161–166, Montpellier, France.

Moot, R. and Retoré, C. (2011). Second order lambda calculus for meaning assembly: on the logical syntax of plurals. In *Coconat*, Tilburg, Pays-Bas.

Moot, R. and Retoré, C. (2012). *The logic of categorial grammars: a deductive account of natural language syntax and semantics*, volume 6850 of *LNCS*. Springer. `http://www.springer.com/computer/theoretical+computer+science/book/978-3-642-31554-1`.

Muskens, R. (1990). Anaphora and the logic of change. In van Eijck, J., editor, *JELIA*, volume 478 of *Lecture Notes in Computer Science*, pages 412–427. Springer.

Nunberg, G. (1995). Transfers of meaning. *Journal of semantics*, 12(2):109–132.

Partee, B. (2008). Noun phrase interpretation and type shifting principles. In Partee, B. and Portner, P., editors, *Formal Semantics: The Essential Readings*, pages 357–381. Wiley.

Peters, S. and Westerståhl, D. (2006). *Quantifiers in Language and Logic*. Clarendon Press.

Pogodalla, S. and Prost, J.-P., editors (2011). *Logical Aspects of Computational Linguistics - 6th International Conference, LACL 2011, Montpellier, France, June 29 - July 1, 2011. Proceedings*, volume 6736 of *LNCS*. Springer.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, 17(4):409–441.

Pustejovsky, J. (1995). *The generative lexicon.* M.I.T. Press.

Real-Coelho, L.-M. and Retoré, C. (2013). A generative montagovian lexicon for polysemous deverbal nouns. In *4th World Congress and School on Universal Logic – Workshop on Logic and linguistics.*

Retoré, C. (2012). Variable types for meaning assembly: a logical syntax for generic noun phrases introduced by "most". *Recherches Linguistiques de Vincennes*, 41:83–102. `http://hal.archives-ouvertes.fr/hal-00677312`.

Soloviev, S. and Luo, Z. (2000). Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic*, 1-3(113):297–322.

Talmy, L. (1999). Fictive motion in language and "ception". In Bloom, P., Peterson, M. A., Nadel, L., and Garrett, M. F., editors, *Language and Space*, pages 211–276. MIT Press.

von Heusinger, K. (1997). Definite descriptions and choice functions. In Akama, S., editor, *Logic, Language and Computation*, pages 61–91. Kluwer.

von Heusinger, K. (2004). Choice functions and the anaphoric semantics of definite nps. *Research on Language and Computation*, 2:309–329.

Xue, T. and Luo, Z. (2012). Dot-types and their implementation. In Béchet and Dikovsky (2012), pages 234–249.